ISG 595A

# Deep Learning for Autonomous Cars – CNN Implementation using Caffe on NVIDIA Jetson TK1

5/2/2017

by
Aravind Krishnan, *akrishnan@wpi.edu*

Advisor
Carlos W Morato, *cwmorato@wpi.edu*

**Independent Study**

**Worcester Polytechnic Institute**

# Table of Contents

# List of Figures

# 1: Introduction

Deep Learning is rapidly advancing and a lot of researchers already are engaged to bring out astounding results in terms of learning methods. These methods are applied to a wide range of fields like market economics, medicine research, perception, robotics, automobile industry and even stem cell research.

My previous project on Deep Learning was on "Empirical Evaluation of Convolutional Neural Networks in Classifying German Traffic Signs". That project was an analysis of the prediction time by 6 CNN models (shallow- 3 CNN layers to deep- 31 CNN layers) in classifying the German Traffic Signs. This was also performed on four different hardware configurations in terms of computing power. The inputs images were of two types:

- 32x32 gray scale images
- 128x128 gray scale images

Results were really helpful in understanding how the computing power played a role and what were the differences with the classification on two types of inputs and the performance of the 6 different CNN models.

My idea was to extend this analysis by conducting the experiments on the NVIDIA Jetson TK1 embedded platform that has 192 GPU cores exclusively for such Deep Learning projects. My idea was also to use the pre-trained 6 different CNN models developed in the previous project along with their weights and just run a test on the Jetson TK1. But unfortunately Keras does not support on TK1 and there is no actual use of a comparison if I am unable to run my models with pre-trained weights on the TK1. So the only choice left was to choose a well-adapted Deep Learning framework like Caffe that has higher rate of working on the Jetson TK1.

## 1.1 Project Statement

The goal of this project is:

1. Review existing Deep Learning frameworks and choose the best that works on Jetson TK1
2. Upgrade the Jetson TK1 to the latest drivers to work with stable releases
3. Install Caffe on the Jetson TK1
4. Run MNIST example successfully as a sanity check
5. Prepare the GTSRB dataset used in the previous project to work with Caffe
6. Plan to modify the previously developed CNN models in the format that Caffe can read

## 1.2 Summary

I want to provide a roadmap that provides the details to be followed to train and test convolutional neural networks that were originally developed under the Theano/Keras Deep Learning frameworks to work on the Caffe framework. These models should be trained and tested on the NVIDIA's Jetson TK1 embedded platform. Though the idea seems possible, there are a lot of dependencies in the 64-bit ARM Ubuntu that runs on the Jetson TK1 that may be a hindrance to this project.

# 2: Background

This report is specifically intended and written for readers who have a background in Deep Learning, especially with classifying traffic signs using convolutional neural networks. However, keeping in mind that some readers may not be familiar with the contents in this report, I present some limited background material in this section.

## 2.1 Software

This section contains background information on the various Deep Learning frameworks and libraries – Caffe, Theano and Keras; that are popular and fits the needs of the task.

### 2.1.1 Caffe Framework

Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by Berkeley AI Research (BAIR) and by community contributors. Yangqing Jia created the project during his PhD at UC Berkeley. [1] LeNet is the Caffe CNN model commonly used on the MNIST dataset to get really high accuracies. Below is the image of a LeNet model.

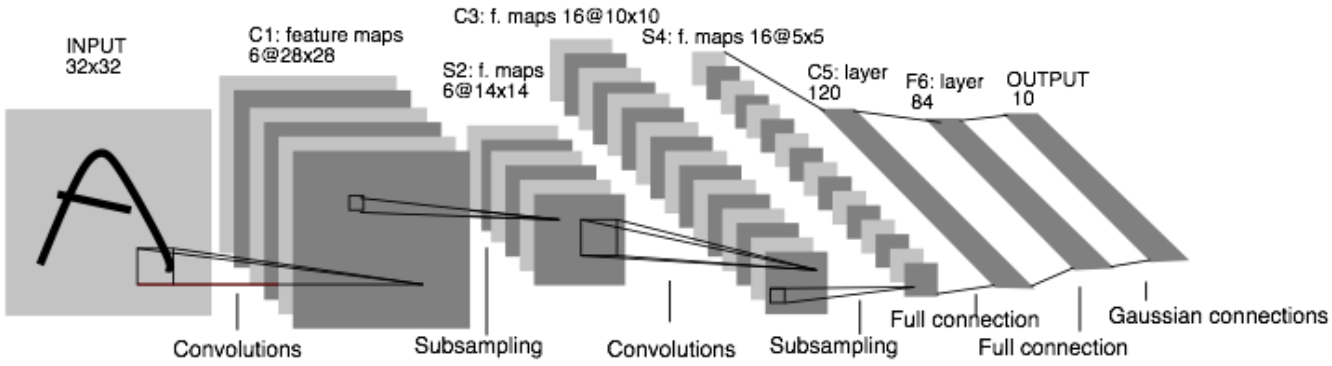


*Figure 1 : LeNet: a layered model composed of convolution and subsampling operations followed by a holistic representation and ultimately a classifier for handwritten digits.: LeNet: a layered model composed of convolution and subsampling operations followed by a holistic representation and ultimately a classifier for handwritten digits.*

The layers in the Caffe Deep Learning framework comprises of blobs and they connect one layer to another. The bottom layer is the data from the dataset and top layer is the convolutional neural network layer.

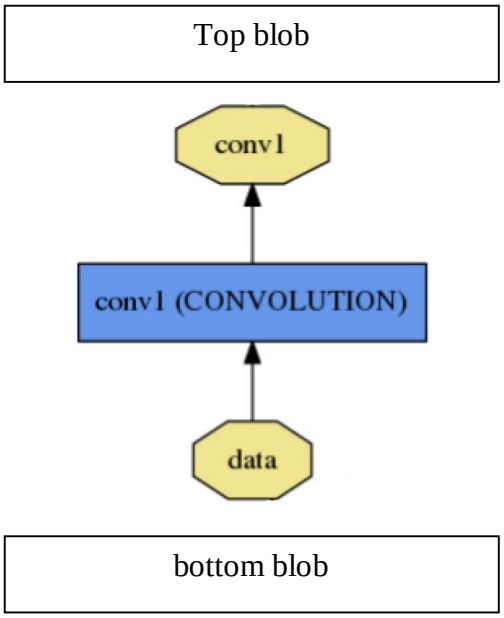


*Figure 2: Visualisation of a blobs in Caffe model*

Blobs are N-D arrays for storing and communicating information between layers. They hold data, derivatives and parameters, allocate memory, shuttle between CPU and GPU.

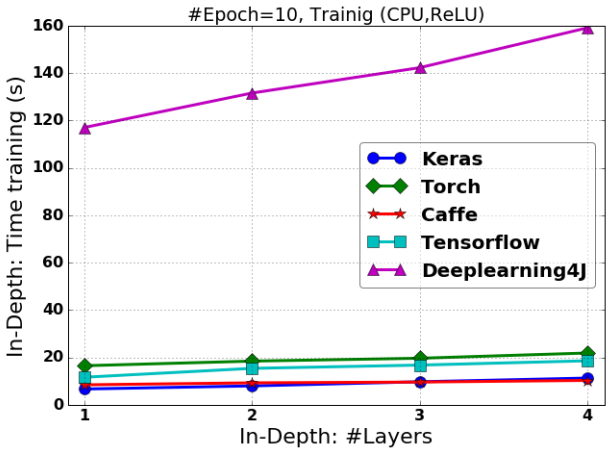


*Figure 3: Comparison between different Deep Learning Frameworks [2]*

The figure above shows a comparison of all the frameworks and it is a plot of number of layers vs training time. The authors in this paper used FCNN with a Tanh nonlinearity function. Caffe clearly shows that it requires very less time than most of the other frameworks.

### 2.1.2 Theano

Theano is a Python library that allows one to define, optimize, and evaluate mathematical expressions, especially ones with multi-dimensional arrays (numpy.ndarray). It helps in faster computations equaling that achieved by C programming language. It can run on Graphical Processing Units which is quite essential for the acceleration of training of neural networks.

Theano combines aspects of a computer algebra system (CAS) with aspects of an optimizing compiler. It can also generate customized C code for many mathematical operations. This combination of CAS with optimizing compilation is particularly useful for tasks in which complicated mathematical expressions are evaluated repeatedly and evaluation speed is critical. For situations where many different expressions are each evaluated once Theano can minimize the amount of compilation/analysis overhead, but still provide symbolic features such as automatic differentiation. [3]

### 2.1.3 Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of either TensorFlow or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.
Keras can be used if one needs a deep learning library that:

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU. [4]

## 2.2 Hardware

This section contains information about the hardware – NVIDIA Jetson Tegra K1 (TK1) that was used as a platform to install Caffe and develop Convolutional Neural Networks to classify the German Traffic Signs and obtain the classification time.

### 2.2.1 NVIDIA Jetson TK1

NVIDIA Jetson Tegra-K1(TK1) is an embedded hardware platform provided by NVIDIA. It is specifically designed for applications involving faster computations on the GPU, such as deep learning where training of the neural networks happens comparatively faster than on a CPU alone. The features of NVIDIA Jetson TK1 are: [5]

- Tegra K1 SOC
- NVIDIA Kepler GPU with 192 CUDA Cores
- NVIDIA 4-Plus-1™ Quad-Core
- ARM® Cortex™-A15 CPU
- 2 GB x16 Memory with 64-bit Width
- 16 GB 4.51 eMMC Memory
- 1 Half Mini-PCIE Slot
- 1 Full-Size SD/MMC Connector
- 1 Full-Size HDMI Port

- 1 USB 2.0 Port, Micro AB
- 1 USB 3.0 Port, A
- 1 RS232 Serial Port
- 1 ALC5639 Realtek Audio Codec with Mic
- In and Line Out
- 1 RTL8111GS Realtek GigE LAN
- 1 SATA Data Port
- SPI 4 MByte Boot Flash

## 2.3 Model and Dataset

### 2.3.1 German Traffic Sign Recognition Benchmark Dataset

The German Traffic Sign Recognition Benchmark Dataset consists of all the traffic signs of Germany. There are about 43 different signs. This is a dataset because it is a collection of multiple images of each traffic sign and there are about 50,000 images in this dataset. This dataset was used in the competition held in the year 2011. And it has since then been a benchmark dataset for the recognition and classification of the traffic signs where a number of Neural networks were used to identify the signs and get the best accuracy. [6]

### 2.3.2 Convolutional Neural Network

Convolutional Neural Networks are very similar to ordinary Neural Networks. They are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels (as input to the network) on one end to class scores (prediction probability of different classes) at the other. And they use a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer. [7]

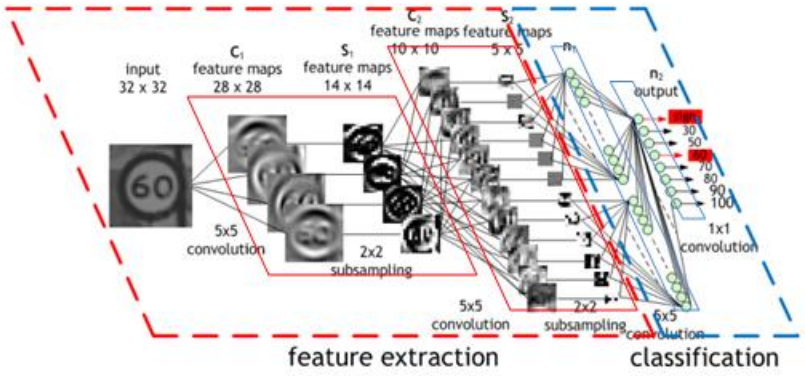


*Figure 4: A CNN with a traffic sign as an input [8]*

# 3: Methods and Approaches

As discussed in the Introduction section, the initial approach was to include the NVIDIA Jetson TK1 as one of the hardware platforms to be compared among various other computationally powerful GPUs from NVIDIA. My previous project was to get the real time prediction and classification of a German traffic sign using CNNs that were self-developed. With the limitations mentioned in the introduction section, I will discuss the methods and approaches that I took to prepare the TK1 and the GTSRB dataset to still be helpful in the hardware comparison.

The methods began with the flashing of NVIDIA Jetson TK1 with the latest 21.5 L4T drivers and Jetpack 2.3.1 version from the official website. Later, the Caffe Deep Learning framework installation on the TK1, MNIST dataset as an example and then steps to prepare my project's GTSRB dataset to work with Caffe.

## 3.1 Jetson TK1 setup

### 3.1.1 Flashing L4T drivers and Jetpack 2.3.1

The latest and official release of L4T drivers for the TK1 is version 21.5 and the supporting Jetpack version is 2.3.1. The TK1 originally comes with an older version of these drivers and it depends on the user whether to upgrade it or not. The detailed instructions of flashing the Jetson TK1 with these drivers are provided in this link: *http://docs.nvidia.com/jetpack-l4t/2_3/index.html#developertools/mobile/jetpack/l4t/2.3/jetpack_l4t_install.htm*

This process is described in detail to help the user easily upgrade the Jetson TK1's OS and drivers. A 64-bit ARM architecture Linux version will be installed on the Jetson TK1. JetPack L4T includes host (Ubuntu Desktop) and target (Jetson Developer Kit) development tools, APIs, and packages (OS images, tools, middleware, samples, and documentation) for developing with NVIDIA® Tegra® on the NVIDIA Jetson Embedded Platform.

*OS Images*

A sample file system derived from Ubuntu for Jetson.

*Libraries*

- CUDA Toolkit for Host (Ubuntu with cross-development support)
- CUDA 8.0 Toolkit for Jetson on L4T
- VisionWorks
- OpenCV4Tegra
- cuDNN
- NVIDIA GPU Inference Engine (GIE)
- MultiMedia API
- Developer Tools

*Tegra System Profiler*

A multi-core CPU sampling profiler that provides an interactive view of captured profiling data, helping improve overall application performance.

*Tegra Graphics Debugger*

A console-grade tool that allows developers to debug and profile OpenGL ES 2.0, OpenGL ES 3.0, OpenGL ES 3.1, OpenGL 4.3, OpenGL 4.4 and OpenGL 4.5, enabling game and graphics developers to get the most out of Tegra.

*PerfKit*

A software library that provides access to OpenGL driver and GPU hardware performance counters.

*Samples*

NVIDIA GameWorks OpenGL Samples

*Documentation*

JetPack Documentation [9]

## 3.2 Caffe Installation

The Caffe installation, surprisingly, was a little unclear on the official Caffe installation website by the Berkeley Vision and Learning Center. After a lot of searching on the internet, I found one person's blog that had neat and simple instructions to get the Caffe installed on the TK1 and it works. It is by a person named Huangying-Zhan and the installation procedure that I followed is given here: *https://huangying-zhan.github.io/2016/08/16/Caffe-installation-and-practice-on-Jetson-TK1.html*

The following steps in brief are necessary to get Caffe installed:

- General dependencies
- CUDA for ARM package
- OpenCV for Tegra
- Installing Caffe – by cloning the official git repository of Caffe
- Run tests to check the Caffe layers
- Install PyCaffe

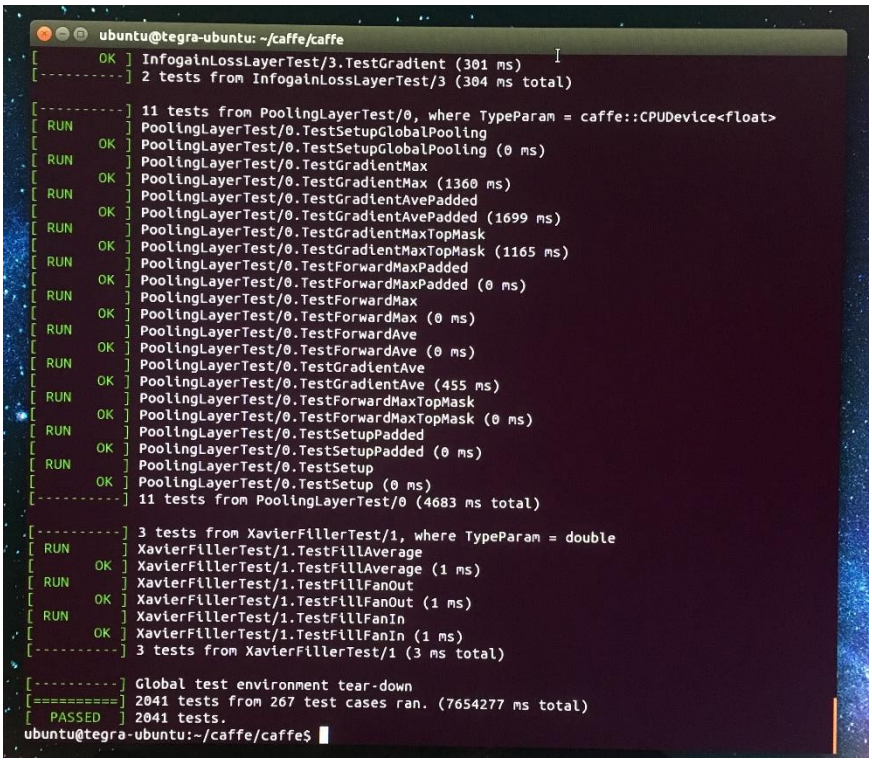


*Figure 5: An image showing the passing of all the tests of the Caffe layers*

## 3.3 MNIST Example

The MNIST is a database of handwritten digits that has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting. [10]

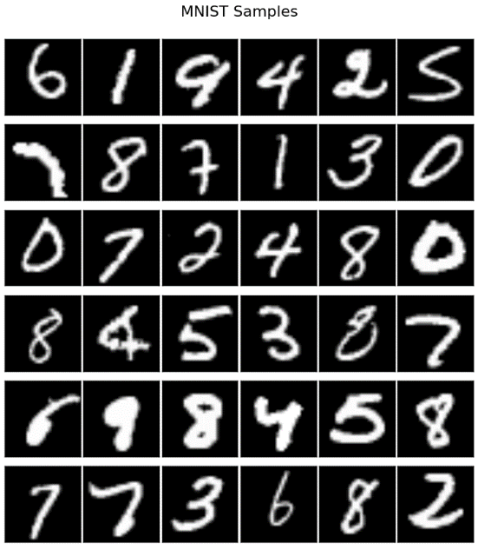


*Figure 6: Sample of the MNIST dataset*

The MNIST dataset was run as an example to check the Caffe's working on the TK1. While training the accuracy was about 99.07% and while testing it was about the same.

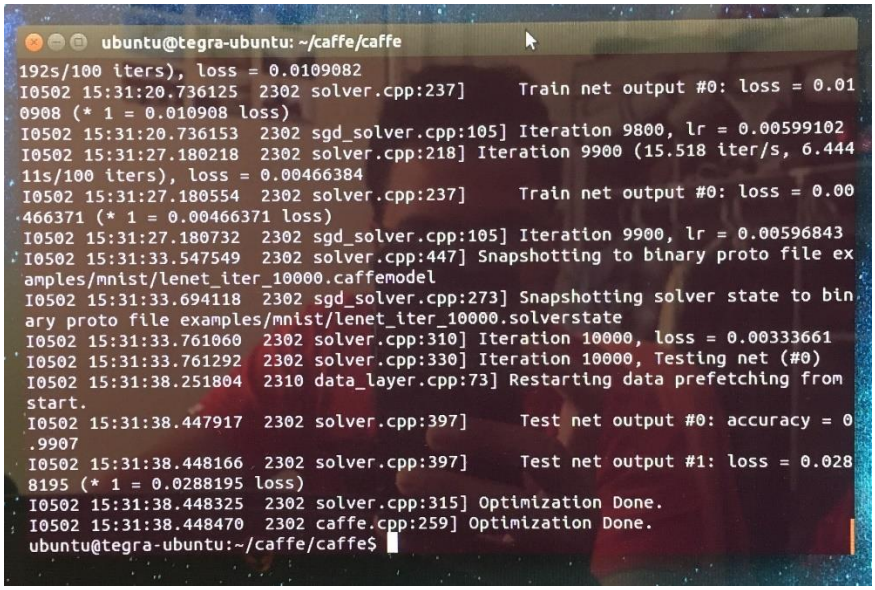


*Figure 7: An image showing the accuracy obtained on the MNIST dataset using the LeNet model*

## 3.4 Preparing the GTSRB dataset

As a part of this project, the preparation of the GTSRB dataset is essential to get it working on the TK1 using Caffe. A person named Marko Arsenovic's github explains one way of doing this. [11] He explains the procedure to create the custom dataset that can be trained using Caffe with the ImageNet model, but the same procedure holds good for any self-developed CNN model in Caffe. The repository contains all the necessary scripts to do the preprocessing and the training necessary before the testing. The steps to be followed are here:

- Make two folders named Test and Train and put all the images of the GTSRB dataset inside the Test and Train folders for test and valuation appropriately. Example, the Test and Train folders must each contain all the 43 classes of traffic signs in individual folders -- /Train/stop; /Train/no-entry; /Test/stop; /Test/no-entry; and the rest.

- Create your own train.txt and val.txt files using train_val.py script. The content of these txt files will be as example:
  - Train/stop/stop.jpg 0 Test/stop/stop.jpg 0
  - Train/no-entry/no-entry.jpg 1 Test/no-entry/no-entry.jpg 1
  - The images are actually named against their class numbers and are the same in the Test and the Train folders.

- The second stage is to create leveldb libraries using create_leveldb.sh, by changing the required paths inside the script.
- After that, compute the mean using make_<yourCNNmodel>_mean.sh, the <yourCNNmodel>_mean.binaryproto file will be created.
- Create proto files, examples are provided, you should have train & val proto files and solver proto file for setting training parameters (momentum, decay, snapshots, no. of iterations etc.). Change the num_output in fully-connected layer for custom number of classes, for example num_output = 43 for the GTSRB dataset because there are 43 traffic signs.
- The next step is to start training the Net, using finetune_<yourCNNmodel>.sh, gpu or cpu, the log of training will be placed in output_finetune.txt.
- After the training is done, Caffemodel files will be generated, snapshots will be saved after every 1000th iteration. (This snapshot interval can be changed in the solver proto file.)
- To test the network, simple python scripts can be written that use the model file, dataset and the solver proto file. There are also scripts to plot the error, accuracies and scores during valuation and testing respectively.

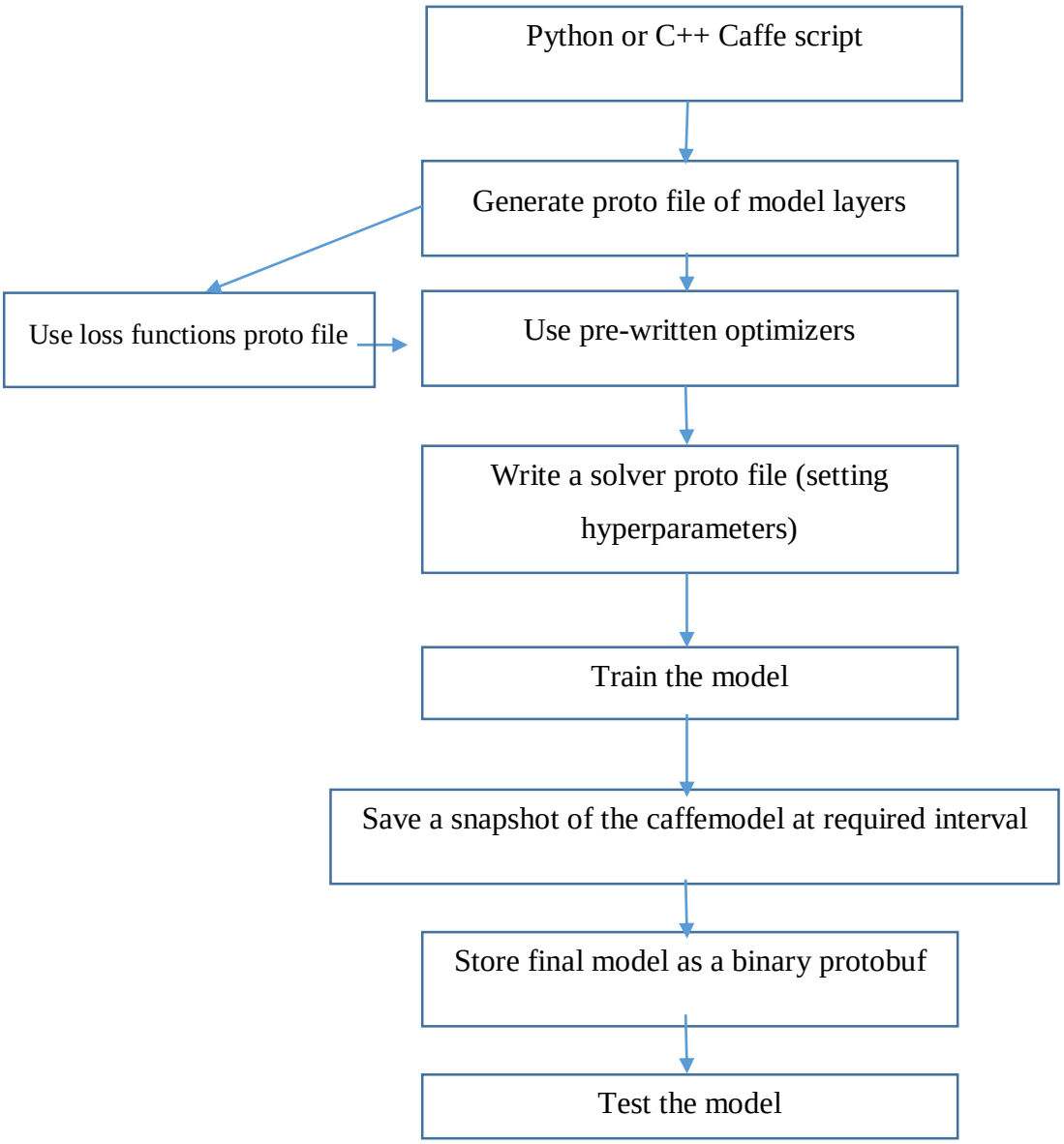


*Figure 8: A flow of the process in writing a model, training and testing it in Caffe*

## 4: Future Work and Conclusion

The following steps must be done in order to get my previous Deep learning project to work on TK1 using Caffe.

- Convert the models file that is written using Keras/Theano in python to a compatible version for Caffe.
- The 6 different CNN models should be written as a prototxt file to work with Caffe.
- The optimizers and loss functions are written in respective proto files again.
- The hyperparameters should be mentioned in the model prototxt file.

- The solver proto file is suggested to be different for the 6 different CNN models which would be provide flexibility during training and testing the models on the GTSRB dataset.

Once all these files are ready, the training can be done on the TK1 but I suspect that it will take relatively a lot of time to train all the 6 models, even after using the GPU, because it has a relatively low computing power when compared to GPU 1070 with 8GiB of RAM by NVIDIA. This process will help us understand how the real time prediction would perform on a platform like the Jetson TK1 which is exclusively developed by Deep Learning purposes and running a Deep Learning framework- Caffe, that is well adapted for this embedded platform.

## 5: References

[1] http://caffe.berkeleyvision.org/
[2] Deep Learning with Theano, Torch, Caffe, TensorFlow, and Deeplearning4J: Which One Is the Best in Speed and Accuracy?, authored byVassili Kovalev, Alexander Kalinovsky and Sergey Kovalev
[3] http://deeplearning.net/software/theano/introduction.html
[4] https://keras.io/
[5] http://www.nvidia.com/object/jetson-tk1-embedded-devkit.html#sthash.aXdxUk9f.dpuf
[6] http://benchmark.ini.rub.de/?section=gtsrb&subsection=news
[7] http://cs231n.github.io/convolutional-networks/
[8] https://devblogs.nvidia.com/wp-content/uploads/2015/11/fig1.png
[9] http://docs.nvidia.com/jetpack-l4t/2_3/index.html#developertools/mobile/jetpack/l4t/2.3/jetpack_l4t_whats_included.htm%3FTocPath%3DJetPack%2520L4T%7C_____2
[10] http://yann.lecun.com/exdb/mnist/
[11] https://github.com/MarkoArsenovic/TrainCaffeCustomDataset